

ISIS-II PL/M-86 V2.0 COMPILATION OF MODULE STAT

OBJECT MODULE PLACED IN STAT.OBJ

COMPILER INVOKED BY: !F0: STAT.PLM DATE(MARCH 16, 1982) XREF OPTIMIZE(3) DEBUG

```

1      $title ('STATUS')
      stat:
      d

```

```

/* common stat module for MP/M-80 2.0 and MP/M-86 2.0
   CP/M-80 2.2 and CP/M-86 1.1 */

```

/*

```

Copyright(c) 1975, 1976, 1977, 1978, 1979, 1980, 1981
Digital Research
Box 579
Pacific Grove, Ca
93950

```

Revised:

```

20 Jan 80 by Thomas Rolander
29 July 81 by Doug Huskey (for MP/M 2.0)
02 Sept 81 (for MP/M-86)
14 Nov 81 by Doug Huskey (for CP/M-86)
16 Dec 81 by Doug Huskey (for CP/M-86)

```

*/

```

/* modified 10/30/78 to fix the space computation */
/* modified 01/28/79 to remove despool dependencies */
/* modified 07/26/79 to operate under cp/m 2.0 */

```

```

#include(copyrt.lit)

```

```

=
= /*
= Copyright (C) 1981
= Digital Research
= P.O. Box 579
= Pacific Grove, CA 93950
= */
=

```

/*

```

Note: In an attempt to have a common source for
all DRI O.S. shared utilities the version is tested
and a number of conditionally executed statements
have been added for compatibility between MP/M 2
and CP/M. This includes handling both R/O (under
CP/M and RO).

```

*/

```

/* commands used to generate 8086 version */

```

CP/M-86 v.1.1 (Vol. II)

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # C81-162

```

/* (on VAX)
asm86 scd.a86
asm86 statex.a86
plm86 stat86.plm 'p2' 'p3' 'p4' optimize(3) debug
plm86 dpb86.plm 'p2' 'p3' 'p4' optimize(3) debug
link86 scd.obj,stat86.obj,statex.obj,dpb86.obj to stat86.lnk
loc86 stat86.lnk od(sm(code,dats,data,stack,const)) -
    ad(sm(code(0)) ss(stack(+32))
h86 stat86

```

```

(on a micro)
vax stat86.h86 $fans
gencmd stat86 datafb167 m352 x800j

```

```

* note the beginning of the data segment will change when
* the program is changed. see the 'MP2' file generated by
* LOC86. also the constants are last to force hex generation
* 352h paragraphs is enough for 512 directory entries
*/

```

```

2 1 declare
    bdos3ver literally '30h'; /* tests for 3.0' cp/m bdos */

```

```

#include(comlit.lit)

```

```

3 1 = declare
    = lit literally 'literally',
    = dcl lit 'declare',
    = proc lit 'procedure',
    = addr lit 'address',
    = true lit 'offh',
    = false lit '0',
    = boolean lit 'byte',
    = forever lit 'while true',
    = cr lit '13',
    = lf lit '10',
    = tab lit '9',
    = ff lit '12',
    = sectorlen lit '128';

```

```

/*****
*
* B D O S INTERFACE
*
*****/

```

```

4 1 mon1:
    procedure (func,info) external;
5 2 declare func byte;
6 2 declare info address;
7 2 end mon1;

8 1 mon2:
    procedure (func,info) byte external;

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

9 2      declare func byte;
10 2     declare info address;
11 2     end mon2;

12 1     mon3:
        procedure (func,info) address external;
13 2     declare func byte;
14 2     declare info address;
15 2     end mon3;

16 1     base$dpb: procedure external;      /* set up base of dpb in dpb86 */
17 2     end base$dpb;                     /* or in dpb80 module      */

18 1     dpb$word: procedure (dpbindex) address external;
19 2     declare dpbindex byte;
20 2     end dpb$word;

21 1     dpb$byte: procedure (dpbindex) byte external;
22 2     declare dpbindex byte;
23 2     end dpb$byte;

        $include(dpb.lit)
        =
        = /* indices into disk parameter block, used as parameters to dpb procedure */
        =
24 1  = dcl      spt$w      lit      '0',
        =          blkshf$b  lit      '2',
        =          blkmask$b lit      '3',
        =          extmask$b lit      '4',
        =          blkmax$w  lit      '5',
        =          dirmax$w  lit      '7',
        =          dirblk$w  lit      '9',
        =          chksiz$w  lit     '11',
        =          offset$w  lit     '13';
        =

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

        /* returns the number of allocated blocks in the
        allocation vector */
25 1     get$all: procedure (dsa) address external;
26 2     declare dsa address;
27 2     end get$all;

```

```

        /* returns the number of CRT columns */
28 1     columns: procedure byte external;
29 2     end columns;

```

```

/* *****

```

```

*** GLOBAL DATA ***

```

```

*****

```

/* BDOS INTERFACE */

```

30 1 declare cmdrv byte external; /* command drive */
31 1 declare fcb (1) byte external; /* 1st default fcb */
32 1 declare fcb16 (1) byte external; /* 2nd default fcb */
33 1 declare pass0 address external; /* 1st password ptr */
34 1 declare len0 byte external; /* 1st passwd length */
35 1 declare pass1 address external; /* 2nd password ptr */
36 1 declare len1 byte external; /* 2nd passwd length */
37 1 declare tbuff (1) byte external; /* default dma buffer */

38 1 declare
    maxb address external; /* addr field of jmp BDOS */
    buff(128) byte external; /* default buffer */
    buffc literally '.buff', /* default buffer */
    fcba literally '.fcb', /* default file control block */
    dolla literally '.fcb(6dh-5ch)', /* dollar sign position */
    parma literally '.fcb(6dh-5ch)', /* parameter, if sent */
    rreca literally '.fcb(7dh-5ch)', /* random record 7d,7e,7f */
    rreco literally '.fcb(7fh-5ch)', /* high byte of random overflow */
    memsize literally 'maxb', /* end of memory */
    rrec address at(rreca), /* random record address */
    rovf byte at(rreco), /* overflow on getfile */
    doll byte at(dolla), /* dollar parameter */
    parm byte at(parma); /* parameter */

```

/* SEARCH AND SEARCH NEXT FCB */

```

39 1 declare
    dcnt byte,
    ufcba byte initial('?'), /* search all dir entries */
    bfcba address, /* index into dir buffer */
    bfcuser based bfcba byte,
    bfcba based bfcba (32) byte; /* template over dirbuff */

```

/* GENERAL GLOBALS */

```

40 1 declare
    fcbmax literally '512', /* max fcb count */
    spkshf literally '3', /* 2**n sectors per kbyte */
    ioval byte, /* io byte */
    sizeset byte initial(0), /* true if displaying size field */
    set$attribute byte initial(0), /* true if setting file attributes */
    user$code byte, /* current user code */
    ver address, /* OS version number */
    kpb address, /* kbytes per block */
    all$blks address, /* number of allocated blocks */
    bdos3 byte, /* is it bdos version 3 ? */
    rodisk address, /* read only disk vector */
    cdisk byte; /* current disk */

```

```

41 1 declare
    byte3 structure (
        lword address,
        hbyte byte);

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P.O. BOX 570

PACIFIC GROVE, CA 93950

/* SCANNER */

```

42 1 declare
    accum(4) byte, /* accumulator */
    ibp byte initial(1); /* input buffer pointer */

```

```

43 1 declare
    drivename ($) byte data
        (' Drive ',0),
    readonly ($) byte data
        ('Read Only (RO)',0),
    readwrite ($) byte data
        ('Read Write (RW)',0),
    system ($) byte data
        ('System (Sys)',0),
    directory ($) byte data
        ('Directory (Dir)',0),
    entries ($) byte data
        (' Directory Entries',0),
    filename ($) byte data
        ('d:filename.typ ',0),
    use ($) byte data
        ('Use: STAT ',0),
    invalid ($) byte data
        ('Invalid Assignment',0),
    setsto ($) byte data
        (' set to ',0),
    record$seg ($) byte data
        ('128 Byte Record',0),
    attributes ($) byte data
        ('[ra] [rw] [sys] or [dir]',0),
    devl ($) byte data
        ('CON:AXI:AXO:LST:DEV:VAL:USR:DSK:'),
    attribl ($) byte data
        ('RO RW SYS DIR SIZE');

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

/* PRINT A 3 BYTE NUMBER */

```

44 1 declare
    val (7) byte initial(0,0,0,0,0,0,0), /* BCD digits */
    fac (7) byte initial(0,0,0,0,0,0,0), /* hi byte factor */
    f0 (7) byte initial(6,3,5,5,6,0,0), /* 65,536 */
    f1 (7) byte initial(2,7,0,1,3,1,0), /* 131,072 */
    f2 (7) byte initial(4,4,1,2,6,2,0), /* 262,144 */
    f3 (7) byte initial(8,8,2,4,2,5,0), /* 524,288 */
    f4 (7) byte initial(6,7,5,8,4,0,1), /* 1,048,576 */
    f5 (7) byte initial(2,5,1,7,9,0,2), /* 2,097,152 */
    f6 (7) byte initial(4,0,3,4,9,1,4), /* 4,194,304 */
    ptr (7) address initial(.f0,.f1,.f2,.f3,.f4,.f5,.f6);

```

/* DISPLAY FILES */

```

45 1  DECLARE      /* duplicate block detection */
      lb byte initial(1),      /* number of bytes per block */
      first$error byte initial(true); /* duplicate block error */

46 1  declare      /* totals */
      kblks address initial(0), /* total number of 1k blks */
      nfcbx address initial(0), /* total number of fcbx */
      nxfcbx address initial(0), /* total number of xfcbx */
      tall address initial(0); /* total allocation */

/* END OF DATA SEGMENT */

47 1  declare
      last$dseg$byte byte initial (0);

48 1  plm: procedure public;

/* *****

*** PRIMITIVE FUNCTIONS ***

*****

49 2  boot: procedure;
50 3      call moni (0,0); /* system reset */
51 3      end boot;

52 2  printchar: procedure(char);
53 3      declare char byte;
54 3      call moni(2,char);
55 3      end printchar;

56 2  printb: procedure;
      /* print blank character */
57 3      call printchar(' ');
58 3      end printb;

59 2  blanks: procedure(b);
60 3      declare b byte;
61 3      do while (b:=b-1) <> -1;
62 4          call printb;
63 4      end;
64 3      end blanks;

65 2  printx: procedure(a);
66 3      declare a address;
67 3      declare s based a byte;
68 3      do while s <> 0;
69 4          call printchar(s);
70 4          a = a + 1;
71 4      end;

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```
72 3      end printx;

73 2      iovalf: procedure byte;
74 3          return mon2(7,0); /* get i/o byte */
75 3      end iovalf;

76 2      sioval: procedure(value);
77 3          declare value byte;
78 3          call mon1(8,value); /* set iobyte */
79 3      end sioval;

80 2      break: procedure byte;
81 3          return mon2(11,0); /* console ready */
82 3      end break;

83 2      test$break: procedure;

84 3          if break then
85 3              do;
86 4                  call mon1(1,0); /* read character */
87 4                  call printx(.(cr,lf,'** Aborted **',0));
88 4                  call boot;
89 4                  end;
90 3          end test$break;

91 2      crlf: procedure;
92 3          call printchar(cr);
93 3          call printchar(lf);
94 3          call test$break;
95 3      end crlf;

96 2      print: procedure(a);
97 3          declare a address;
          /* print the string starting at address a until the
          next 0 is encountered */
98 3          call crlf;
99 3          call printx(a);
100 3      end print;

101 2      version: procedure address;
          /* returns current cp/m version */
102 3          return mon3(12,0);
103 3      end version;

104 2      getrodisk: procedure address;
          /* get the read-only disk vector */
105 3          return mon3(29,0);
106 3      end getrodisk;

107 2      select: procedure(d);
108 3          declare d byte;
109 3          rodisk = getrodisk; /* read only disk vector set */
110 3          cdisk = d;
111 3          call mon1(14,d);
112 3      end select;

113 2      open: procedure(fcb);
```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```
114 3      declare fcb address;
115 3      dcnt = mon2(15,fcb);
116 3      end open;

117 2      search: procedure(fcb);
118 3          declare fcb address;
119 3          declare fcb0 based fcb byte;
120 3          dcnt = mon2(17,fcb);
121 3          end search;

122 2      searchn: procedure;
123 3          dcnt = mon2(18,0);
124 3          end searchn;

125 2      cselect: procedure byte;
           /* return current disk number */
126 3          return mon2(25,0);
127 3          end cselect;

128 2      setdma: procedure(dma);
129 3          declare dma address;
130 3          call mon1(26,dma);
131 3          end setdma;

132 2      getlogin: procedure address;
           /* get the login vector */
133 3          return mon3(24,0);
134 3          end getlogin;

135 2      writeprot: procedure;
           /* write protect the current disk */
136 3          call mon1(28,0);
137 3          end writeprot;

138 2      setind: procedure(a);
139 3          declare a address;
           /* set file indicators for current fcb */
140 3          call mon1(30,a);
141 3          end setind;

142 2      getuser: procedure byte;
           /* return current user number */
143 3          return mon2(32,0ffh);
144 3          end getuser;

145 2      setuser: procedure(user);
146 3          declare user byte;
147 3          call mon1(32,user);
148 3          end setuser;

149 2      getfilesize: procedure(fcb);
150 3          declare fcb address;
151 3          call mon1(35,fcb);
152 3          end getfilesize;
```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____


```

153 2  getfreesp: procedure(d);
154 3      declare d byte;

155 3      call monl(46,d);
156 3      end getfreesp;

```

```

/* we want kpb (Kbytes per block) so that each time we find
a block address we can add kpb k to the kilobyte accumulator
for file size.  e derive kpb as follows:

```

BLKSHF	RECS/BLK	K/BLK	BLKSHF-3
3	8	1	0
4	16	2	1
5	32	4	2
6	64	8	3
7	128	16	4

```

*/

```

```

157 2  set$kp: procedure;
158 3      declare kshf byte;

159 3      call base$dpb; /* disk parameters set */
160 3      if (kshf:=dpb$byte(blkshf$b)-spkshf) < 1 then
161 3          kpb = 1;
          else
162 3          kpb = shl(double(1),dpb$byte(blkshf$b)-spkshf);
163 3      end set$kp;

164 2  select$disk: procedure(d);
165 3      declare d byte;
          /* select disk and set kpb */
166 3      call select(d);
167 3      call set$kp; /* bytes per block */
168 3      end select$disk;

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

/* *****

```

```

*** SCANNER ***

```

```

*****

```

```

          /* fill string @ s for c bytes with f */
169 2  fill: proc(s,f,c);
170 3      dcl (s,c) addr,
          f byte,
          a based s byte;

171 3      do while (c:=c-1) < 0ffffh;
172 4          a = f;
173 4          s = stl;

```

```

174 4      end;
175 3      end fill;
/* ----- */

```

```

/* get next input value into accum */
176 2      scan: procedure;
/* fill accum with next input value */
177 3      declare (i,b) byte;
178 3      setacc: procedure(b);
179 4          declare b byte;
180 4          if i < 4 then
181 4              accum(i) = b;
182 4              i = i + 1;
183 4          end setacc;

```

```

/* deblank input */
184 3      do while (b:=buff(ibp)) = ' ';
185 4          ibp = ibp + 1;
186 4      end;
/* skip option delimiter */
187 3      if b = '[' then
188 3          ibp = ibp + 1;
/* initialize accum length (0-3) */
189 3      i = 0;
190 3      call fill(accum,' ',4);
191 3      do while (b := buff(ibp)) > 1;
192 4          if b < '!' or b = ',' or b = ':' or
            b = '[' or b = '=' then buff(ibp) = 1;
            else
194 4              ibp = ibp + 1;
195 4              if b < '/' and b < '$' and b < ']' and b < ','
                then call setacc(b);
            end;
197 4          end;
198 3      if b < 0 then
199 3          ibp = ibp + 1;
200 3      end scan;
/* ----- */

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

/* parse an assignment into accumulator */
201 2      parse$assign: procedure byte;
202 3          decl b byte;

203 3          call scan;
204 3          if accum(0) < '=' then
205 3              return false;
206 3          call scan;
207 3          return true;
208 3          end parse$assign;
/* ----- */

```

```

/* parse next item into accumulator */
209 2      parse$next: procedure byte;

210 3          call scan;

```

```

211 3      if accum(0) = ' ' then do;
213 4          call scan;
214 4          if accum(0) = ' ' then
215 4              return false;
216 4          end;
217 3      return true;
218 3      end parse$next;
/* ----- */

```

```

/* print a decimal number from 2 byte binary
   in a fixed field (precision) */
219 2      pdecimal: procedure(v,prec);
/* print value v with precision prec (10,100,1000)
   with leading zero suppression */
220 3      declare
          v address, /* value to print */
          prec address, /* precision */
          zerosup byte, /* zero suppression flag */
          d byte; /* current decimal digit */
221 3      zerosup = true;
222 3      do while prec < 0;
223 4          d = v / prec; /* get next digit */
224 4          v = v mod prec; /* get remainder back to v */
225 4          prec = prec / 10; /* ready for next digit */
226 4          if prec < 0 and zerosup and d = 0 then call printb; else
228 4              do; zerosup = false; call printchar('0'+d);
231 5              end;
232 4          end;
233 3      end pdecimal;

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

/* ***** */

*** PRINT A NUMBER ***

***** */

```

/* BCD - convert 16 bit binary to
   7 one byte BCD digits */
234 2      bcd: procedure(value);
235 3      declare
          (value,prec) address,
          i byte;

236 3      prec = 10000;
237 3      i = 5; /* digits: 4,3,2,1,0 */
238 3      do while prec <> 0;
239 4          val(i:=i-1) = value / prec; /* get next digit */
240 4          value = value mod prec; /* remainder in value */
241 4          prec = prec / 10;
242 4          end;
243 3      end bcd;

```

```

/* ----- */

                                /* print BCD number in val array */
244 2  printbcd: procedure(fixed);
245 3  declare
      (fixed, zerosup, i) byte;

246 3  pchar: procedure(c);
247 4  declare c byte;
248 4  if val(i) = 0 then
249 4  if zerosup then
250 4  if i < 0 then do;
252 5  if fixed then
253 5  call printb;
254 5  return;
255 5  end;

      /* else */
256 4  call printchar(c);
257 4  zerosup = false;
258 4  end pchar;

259 3  zerosup = true;
260 3  i = 7;
261 3  do while (i:=i-1) < -1;
262 4  call pchar('0'+val(i));
263 4  if i = 6 or i = 3 then
264 4  call pchar(',');
265 4  end;
266 3  end printbcd;

/* ----- */

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER # _____

```

                                /* add two BCD numbers result in second */
267 2  add: procedure(ap,bp);
268 3  declare
      (ap,bp)      address,
      a based ap (7) byte,
      b based bp (7) byte,
      (c,i)        byte;

269 3  c = 0;                                /* carry */
270 3  do i = 0 to 6;                        /* 0 = LSB */
271 4  b(i) = a(i) + b(i) + c;
272 4  c = b(i) / 10;
273 4  b(i) = b(i) mod 10;
274 4  end;
275 3  end add;

/* ----- */

```

```

                                /* print 3 byte value based at byte3adr */
276 2  p3byte: procedure(byte3adr,fixed);
277 3  declare
      fixed      byte,
      i          byte,
      high$byte byte,

```

```

        byte3adr address,
        b3 based byte3adr structure (
            lword address,
            hbyte byte);

278 3      call fill(.val,0,7);
279 3      call fill(.fac,0,7);
280 3      call bcd(b3.lword);          /* put 16 bit value in val */
281 3      high$byte = b3.hbyte;
282 3      do i = 0 to 6;                /* factor for bit i */
283 4          if high$byte then        /* LSB is 1 */
284 4              call add(ptr(i),.fac); /* add in factor */
285 4              high$byte = shr(high$byte,1); /* get next bit */
286 4          end;
287 3      call add(.fac,.val);          /* add factor to value */
288 3      call printbcd(fixed);         /* print value */
289 3      end p3byte;

/* ----- */

                                /* divide 3 byte value by 8 */
290 2      shr3byte: procedure(byte3adr);
291 3          dcl byte3adr address,
                b3 based byte3adr structure (
                    lword address,
                    hbyte byte),
                temp1 based byte3adr (2) byte,
                temp2 byte;

292 3          temp2 = ror(b3.hbyte,3) and 11100000b; /* get 3 bits */
293 3          b3.hbyte = shr(b3.hbyte,3);
294 3          b3.lword = shr(b3.lword,3);
295 3          temp1(1) = temp1(1) or temp2; /* or in 3 bits from hbyte */
296 3          end shr3byte;

/* ----- */

                                /* multiply 3 byte value by #records per block */
297 2      shl3byte: procedure(byte3adr);
298 3          dcl byte3adr address,
                b3 based byte3adr structure (
                    lword address,
                    hbyte byte),
                temp1 based byte3adr (2) byte;

299 3          b3.hbyte = (rol(temp1(1),dpb$byte(blkshf$b)) and dpb$byte(blkask$b))
300 3              or shl(b3.hbyte,dpb$byte(blkshf$b));
301 3          b3.lword = shl(b3.lword,dpb$byte(blkshf$b));
301 3          end shl3byte;

/* ----- */

                                /* display current drive */
302 2      show$drv: procedure;
303 3          call printchar(cdisk+'A');
304 3          call printchar(':');
305 3          end show$drv;

```

COPYRIGHT 1981, 1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. =

```

/* ----- */

/* display current drive with space */
306 2 show$drive: procedure;
307 3   call show$drv;
308 3   call printb;
309 3   end show$drive;
/* ----- */

```

```

/* print user number */
310 2 show$usr: procedure(user);
311 3   dcl user byte;

312 3   call printx(,('User :',0));
313 3   call pdecimal(user,100);
314 3   end show$usr;

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER # _____

/* ***** */

*** CP/M 2 BLOCK COUNTING ROUTINES ***

/* ***** */

```

/* add # of kilobytes in one block
   kpb = kilobytes per block */
315 2 add$block: procedure(ak);
316 3   declare ak address;
/* add one block to the kilobyte accumulator */
317 3   declare kaccum based ak address; /* kilobyte accum */

318 3   kaccum = kaccum + kpb;
319 3   end add$block;
/* ----- */

```

```

/* returns the number of K remaining */
320 2 count: procedure address;
321 3   declare
       ka address, /* kb accumulator */
       (i, maxall) address; /* local index */
322 3   ka = 0;
323 3   all$blks = getall(dpb$word(blkmax$w));
324 3   if (maxall:=dpb$word(blkmax$w)) > all$blks then
325 3     do i = 0 to maxall-all$blks;
326 4     call add$block(,ka);
327 4     end;
328 3   return ka;
329 3   end count;

```

```
/* *****
```

```
*** STATUS ROUTINES ***
```

```
******/
```

```
/* display drive characteristics */
```

```
330 2 drivestatus; procedure;
331 7     dcl b3a address,
        b3 based b3a structure (
            lword address,
            hbyte byte);

        /* print 3 byte value */
332 3     pv3; procedure;
333 4         call crlf;
334 4         call p3byte(b3a,true);
335 4         call printchar(' ');
336 4         call printb;
337 4         end pv3;

        /* print address value v */
338 3     pv; procedure(v);
339 4         dcl v address;
340 4         b3.hbyte = 0;
341 4         b3.lword = v;
342 4         call pv3;
343 4         end pv;
```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. B. X 579

PACIFIC GROVE, CA 93950

SER. # _____

```
/* print the characteristics of the currently selected drive */
344 3     b3a = .byte3;
345 3     call print(.( '      ',0));
346 3     call show$drive;
347 3     call printx(.( 'Drive Characteristics',0));
348 3     b3.hbyte = 0;
349 3     b3.lword = dpb$word(blkmax$w) + 1; /* # blocks */
350 3     call shl3byte(b3a); /* # blocks * records/block */
351 3     call pv3;
352 3     call printx(.(record$msg);
353 3     call printx(.( ' Capacity',0));
354 3     call shr3byte(b3a); /* divide by 8 */
355 3     call pv3;
356 3     call printx(.( 'Kilobyte Drive Capacity',0));
357 3     call pv(dpb$word(dirmax$w)+1);
358 3     call printx(.( '32 Byte',0));
359 3     call printx(.(entries);
360 3     call pv(shl(dpb$word(chksiz$w),2));
361 3     call printx(.( 'Checked',0));
362 3     call printx(.(entries);
363 3     call pv((dpb$byte(extask$b)+1) * 128);
364 3     call printx(.(record$msg);
```

```

365 3      call printx(,('s / Directory Entry',0));
366 3      call pv(shl(double(1),dph$byte(blkshf$b)));
367 3      call printx(,record$msg);
368 3      call printx(,('s / Block',0));
369 3      call pv(dph$word(spt$w));
370 3      call printx(,record$msg);
371 3      call printx(,('s / Track',0));
372 3      call pv(dph$word(offset$w));
373 3      call printx(' (Reserved Tracks',0));
374 3      call crlf;
375 3      end drivestatus;

```

```
/* ----- */
```

```

                                /* display active user numbers */
376 2      userstatus: procedure;
                                /* display active user numbers */
377 3      declare i byte;
378 3      declare user(16) byte;
379 3      call crlf;
380 3      call show$drive;
381 3      call printx(,('Active ',0));
382 3      call show$usr(user$code);
383 3      call crlf;
384 3      call show$drive;
385 3      call printx(,('Active Files:',0));
386 3      do i = 0 to last(user);
387 4          user(i) = false;
388 4      end;
389 3      call setdma(.memory);
390 3      call search(.ufcb);
391 3      do while dcnt < 255;
392 4          if (i := memory(shl(dcnt and 11b,5))) < 0e5h then
393 4              user(i and 0fh) = true;
394 4          call searchn;
395 4          end;
396 3      do i = 0 to last(user);
397 4          if user(i) then call pdecimal(i,100);
398 4          end;
400 3      call crlf;
401 3      end userstatus;

```

```
/* ----- */
```

```

                                /* display status of logged in diskettes */
402 2      diskstatus: procedure;
                                /* display disk status */
403 3      declare login address, d byte;
404 3      login = getlogin; /* login vector set */
405 3      d = 0;
406 3      do while login < 0;
407 4          if low(login) then
408 4              do; call select$disk(d);
409 4              call drivestatus;
410 5              end;
411 5          login = shr(login,1);
412 4          d = d + 1;
413 4

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____


```

414 4      end;
415 3      end diskstatus;
/* ----- */

```

```

/* try to match 4 character device in
   accum with device in list
   va = address of list
   vl = # devices in list */
416 2  match: procedure(va,vl) byte;
      /* return index+1 to vector at va if match */
417 3  declare va address,
      v based va (16) byte,
      vl byte;
418 3  declare (i,j,match,sync) byte;
419 3  j, sync = 0;
420 3  do sync = 1 to vl;
421 4  match = true;
422 4  do i = 0 to vl;
423 5  if v(j) <> accum(i) then match=false;
425 5  j = j + 1;
426 5  end;
427 4  if match then return sync;
429 4  end;
430 3  return 0; /* no match */
431 3  end match;
/* ----- */

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

/* process a device request valid devices
   are listed above */
432 2  devreq: procedure byte;
      /* process device request, return true if found */
      /* device tables */
433 3  declare
      devr(2) byte data
      /* console */ 'TTY:CRT:BAT:UC1:',
      /* reader */ 'TTY:PTR:UR1:UR2:',
      /* punch */ 'TTY:PTP:UP1:UP2:',
      /* listing */ 'TTY:CRT:LPT:UL1:');
434 3  declare
      (i,j,iobyte,items,i oval) byte;
435 3  pname: procedure(a);
436 4  declare a address,
      x based a byte;
      /* print device name at a */
437 4  do while x <> '!';
438 5  call printchar(x); a=a+1;
440 5  end;
441 4  call printchar('!');
442 4  end pname;
443 3  devstatus: proc;
444 4  iobyte = i oval; j = 0;

```

```

446 4      do i = 0 to 3;
447 5      call pname(.devl(shl(i,2)));
448 5      call printx(,(' is ',0));
449 5      call pname(.devr(shl(iobyte and 11b,2)+j));
450 5      j = j + 16; iobyte = shr(iobyte,2);
452 5      call crlf;
453 5      end;
454 4      end devstatus;

455 3      values: procedure;
456 4      call printx(,('STAT 2.2',0));
457 4      call crlf;
458 4      call print(,('Read Only Disk: d:=RO',0));
459 4      call print(,('Set Attribute: ',0));
460 4      call printx(.filename);
461 4      call printx(.attributes);
462 4      call print(,('Disk Status : DSK: d:DSK:',0));
463 4      call print(,('User Status : USR: d:USR:',0));
464 4      call print(,('Iobyte Assign:',0));
465 4      do i = 0 to 3; /* each line shows one device */
466 5      call crlf;
467 5      call pname(.devl(shl(i,2)));
468 5      call printx(,(' = ',0));
469 5      do j = 0 to 12 by 4;
470 6      call printchar(' ');
471 6      call pname(.devr(shl(i,4)+j));
472 6      end;
473 5      end;
474 4      call crlf;
475 4      end values;

476 3      items = 0;
477 3      do forever;
478 4      if (i:=match(.devl,8)) = 0 then do;
480 5      if items<0 then
481 5      go to error;
482 5      return false;
483 5      end;
484 4      items = items+1; /* found first/next item */
485 4      if i = 5 then /* device status request */
486 4      call devstatus;
487 4      else if i = 6 then /* list possible assignment */
488 4      call values;
489 4      else if i = 7 then /* list user status values */
490 4      call userstatus;
491 4      else if i = 8 then /* show the disk device status */
492 4      call diskstatus;
493 4      else /* scan item i-1 in device table */
494 5      do; /* find base of destination */
495 5      j = shl(i:=i-1,4);
496 5      if not parse$assign then
497 5      go to error;
498 5      if (j:=match(.devr(j),4)-1) = 255 then
499 5      go to error;
500 5      iobyte = 1111$1100b; /* construct mask */
501 6      do while (i:=i-1) < 255;
502 6      iobyte = rol(iobyte,2);

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

502 3          j = shl(j,2);
503 3          end;
504 3          ioval = (iovalf and iobyte) or j;
505 3          call siovalf(ioval);
506 3          end;
          /* end of current item, look for more */
507 3          if not parse$next then
508 3              return true;
509 3          end; /* of do forever */
510 3 error:
          call print(.invalid);
511 3          return true;
512 3          end devreq;
/* ----- */

```

```

          /* print the actual byte count */
513 2 prcount: procedure(fixed);
514 3     declare fixed byte;

515 3     if bdos3 then do;
517 4         call setdma(.byte3);
518 4         call getfreesp(cdisk);
519 4         call shr3byte(.byte3);
520 4         end;
521 3     else do;
522 4         byte3.hbyte = 0;
523 4         byte3.lword = count;
524 4         end;
525 3     call p3byte(.byte3,fixed);
526 3     call printchar('K');
527 3     end prcount;
/* ----- */

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

          /* print allocation for current disk */
528 2 pralloc: procedure;
529 3     dcl
          ro address;

530 3     call crlf;
531 3     call show$drive;
532 3     call printchar('R');
533 3     if cdisk <> 0 then
534 3         ro = shr(rodisk,cdisk);
          else
535 3         ro = rodisk;
536 3         if low(ro) then
537 3             call printchar('0');
          else
538 3             call printchar('W');
539 3             call printx((', Free Space: ',0));
540 3             call prcount(true);
541 3             end pralloc;
/* ----- */

```

```

                    /* print the status of the disk system */
542 2  prstatus: procedure;
543 3      declare login address;
544 3      declare d byte;
545 3      login = getlogin; /* login vector set */
546 3      d = 0;
547 3      do while login <> 0;
548 4          if low(login) then
549 4              do;
550 5                  call select$disk(d);
551 5                  call pralloc;
552 5                  end;
553 4              login = shr(login,1);
554 4              d = d + 1;
555 4              end;
556 3      call crlf;
557 3      end prstatus;

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

/* ***** */

*** DISPLAY FILES ***

***** */

```

                    /* process file display or attribute
                    assignment */
558 2  getfile: procedure;
                    /* process file request */
559 3  declare
        fnam  literally '11',
        fext  literally '12',
        fs1   literally '13',
        fmod  literally '14',
        frc   literally '15',
        ftyp  literally '9',
        rofile literally '9', /* read/only file */
        infile literally '10', /* invisible file */
        archiv literally '11', /* archived file */
        attrb1 literally '1', /* attribute F1' */
        attrb2 literally '2', /* attribute F2' */
        attrb3 literally '3', /* attribute F3' */
        attrb4 literally '4'; /* attribute F4' */

560 3  declare
        membase address, /* base for collecting fcb's */
        fcb's based membase (1) byte,
        fcbn address, /* number of fcb's collected so far */
        finx(fcbmax) address, /* index vector used during sort */
        fchr(fcbmax) address, /* record count */
        fcb$ address, /* index into fcb's */

```

```

fcbca address, /* ext byte of fcbv */
fcbci address, /* si byte of fcbv */
fcbca address, /* mod byte of fcbv */
fcbca based fcbca address, /* extent count in fcbs entry */
fcbk based fcbca address, /* kbyte count in fcbs entry */
xpcb based fcbci byte, /* bit 7 = xpcb for fcbs entry */
fcbv based fcbca (16) byte; /* template over fcbs entry */

```

```

561 3 declare
      i address, /* fcb counter during collection and display */
      l address, /* used during sort and display */
      k address, /* " */
      m address, /* " */
      kb byte, /* byte counter */
      lb byte, /* byte counter */
      mb byte, /* byte counter */
      (b,f) byte, /* counters */
      xpcbfound byte, /* true means xpcb found */
      wordblks byte, /* 2 byte block addresses */
      matched byte; /* used during fcbs search */

```

```

562 3 declare
      (scase1,scase2) byte; /* status case # */
/* ----- */

```

```

/* display a line of b dashes */
563 3 dots: procedure(i);
564 4 declare i byte;

565 4 call crlf;
566 4 do while (i:=i-1) < -1;
567 5 call printchar('--');
568 5 end;
569 4 end dots;
/* ----- */

```

```

/* print current file name */
570 3 printfn: procedure(a);
571 4 declare

```

```

      a address,
      fcbv based a (16) byte,
      (k, lb) byte;

572 4 call show$drv;
573 4 do k = 1 to fnam;
574 5 if k = ftyp then
575 6 call printchar(',');
576 6 call printchar(fcbv(k) and 7fh);
577 6 end;
578 4 end printfn;
/* ----- */

```

```

/* test if attribute fcbv(i) is on */
579 3 attribute: procedure(i) byte;

```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

```

580 4      declare i byte;

581 4      if rol(fcbv(i),1) then
582 4          return true;
583 4      return false;
584 4      end attribute;

/* ----- */

/* print character c if attribute(b) is true */
585 3      prnt$attrib: procedure(b,c);
586 4          declare (b,c) byte;

587 4      if attribute(b) then
588 4          call printchar(c);
      else
589 4          call printb;
590 4      end prnt$attrib;

/* ----- */

/* build an allocation vector check for duplicate
   blocks in the directory
   return true if no error, false otherwise */
591 3      allocate: procedure(block$ptr) byte;
592 4          declare
              block$ptr address,
              block$word based block$ptr address,
              block$byte based block$ptr byte,
              (vbyte, vbit, amask)      byte;

593 4      if word$blks then do;
595 5          vbyte = shr(block$word,3);
596 5          vbit = block$word mod 8;
597 5          end;
598 4      else do;
599 5          vbyte = shr(block$byte,3);
600 5          vbit = block$byte mod 8;
601 5          end;
602 4      amask = 1000$0000b;
603 4      if vbit < 0 then
604 4          amask = shr(1000$0000b,vbit);
605 4      if (amask and memory(vbyte)) = amask then do;
607 5          if first$error then do;
609 6              first$error = false;
610 6              call print(.( 'Bad Directory on ',0));
611 6              call show$drv;
612 6              call print(.( 'Space Allocation Conflict:',0));
613 6              end;
614 5          call crlf;
615 5          call show$usr(bfcbuser);
616 5          call blanks(8);
617 5          call printfn(bfcbu);
618 5          return false;
619 5          end;
620 4      memory(vbyte) = memory(vbyte) or amask;
621 4      return true;

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. B. X 579

PACIFIC GROVE, CA 93950

SER # _____

```

622 4      end allocate;
/* ----- */

/* check for matching file names */
623 3      file$match: procedure(a) byte;
624 4      declare
            i byte,
            address,
            fcb based a (16) byte;

625 4      do i=1 to 11;
626 5      if fcb(i) <> '?' then
627 5          if (fcb(i) and 7fh) <> (bfcb(i) and 7fh) then
628 5              return false;
629 5      end;
630 4      return true;
631 4      end file$match;
/* ----- */

/* examine block numbers :
   alloc = true => check for duplicate block numbers
   alloc = false => count kbytes mapped by fcb */
632 3      count$blks: procedure(alloc);
633 4      dcl (i,alloc,no$error) byte;
634 4      no$error = true;
635 4      i = 32;
636 4      do while no$error and ((i:=i-1b) >= 16);
637 5      mb = bfcb(i);
638 5      if word$blks then /* double precision inx */
639 5          mb = mb or bfcb(i+1);
640 5      if mb <> 0 then do; /* allocated */
642 6          if alloc then
643 6              no$error = allocate(.bfcb(i));
        else
644 6              fcbk = fcbk + kpb; /* kpb = kbytes per block */
645 6          end;
646 5      end;
647 4      end count$blks;
/* ----- */

/* check next fcb in search / searchn */
648 3      check$user: procedure;
649 4      dcl i byte;
650 4      do forever;
651 5      if dcnt = 255 then
652 5          return;
653 5      bfcbn = shl(dcnt,5)+buffa;
654 5      if bfcbuser < 20h then do;
656 6          call count$blks(true); /* check for duplicate block */
657 6          if file$match(.fcb) then
658 6              if (bfcbuser and 0fh) = user$code
                then return; /* pick up xfcbn and fcbn */
        end;
660 3      end;
661 5      call searchn;

```

COPYRIGHT - 1981, 1982

DIGITAL RESEARCH

P.O. BOX 573

FACIFIC GROVE, CA 93950

SER. =

```

662 5      end;
663 4      end check$user;
/* ----- */

/* set fcbsa to the ith 16 byte fcb in memory
   set fcbca to the extent byte of that fcb */
664 3      multil6: procedure;
/* utility to compute fc : address from i */
665 4      fcbca = (fcbsa:=shl(i,4)+membase) + fext;
666 4      fcbma = fcbca + fmod;
667 4      fcbsl = fcbca + fsl;
668 4      end multil6;
/* ----- */

/* parse file attribute assignment */
669 3      setfilestatus: procedure byte;
/* eventually, scase1 & scase2
   set to r/o=1, r/w=2, dir=3, sys=4, size=5
   scase1 is first attribute assignment
   scase2 is second attribute assignment */

670 4      error: procedure;
671 5          call print(.invalid);
672 5          call print(.use);
673 5          call printx(.filename);
674 5          call printx(('[size] ',0));
675 5          call printx(.attributes);
676 5          end error;

677 4      if not parse$next then
678 4          return false;
679 4      if accum(0) = '=' then
680 4          call scan;
681 4      if (scase1 := match(.attrib1,5)) = 5 then
682 4          return not (sizeset := true);
/* must be a parameter */
683 4      if scase1 = 0 then
684 4          call error;
685 4      else if parse$next then
686 4          if (scase2 := match(.attrib1,5)) = 0 then
687 4              call error;
        return true;
        end setfilestatus;
/* ----- */

/* set the attribute bits */
690 3      set$attributes: procedure(scase);
691 4          declare scase byte;

692 4          do case scase;
/* set to r/o */
693 5              do;
694 6                  fcbv(rofile) = fcbv(rofile) or 80h;
695 6                  call printx(.readonly);

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____


```

696 3      end;
        /* set to r/w */
697 5      do;
698 6          fcbv(rofile) = fcbv(rofile) and 7fh;
699 6          call printx(.readwrite);
700 6      end;
        /* set to sys */
701 5      do;
702 6          fcbv(infile) = fcbv(infile) or 80h;
703 6          call printx(.system);
704 6      end;
        /* set to dir */
705 5      do;
706 6          fcbv(infile) = fcbv(infile) and 7fh;
707 6          call printx(.directory);
708 6      end;
        /* invalid case */
709 5      do;
710 6          call print(.invalid);
711 6          return;
712 6      end;
713 5      end;
714 4      end set$attributes;

```

```
/* ----- */
```

```

        /* check for drive status assignment */
715 3      compare$fcbl: proc;
716 4          matched = false; i = 0;
718 4          do while not matched and i < fcbn;
        /* compare current entry */
719 5          call multil6;
720 5          matched = file$match(fcb$sa);
721 5          i = i + 1;
722 5          end;
723 4          end compare$fcbl;

```

```
/* ----- */
```

```

        /* copy next fcb into fcb$ array */
724 3      copy$fcbl: proc;
        /* copy to new position in fcb$ */
725 4          fcbn = (i := fcbn) + 1;
726 4          call multil6;
        /* fcb$sa set to next to fill */
727 4          if (fcbn > fcb$max) or (fcb$sa + 16) >= memsize then
728 4              do;
729 5                  call print(('.Too Many Files',0));
730 5                  call boot; /* abort */
731 5              end;
        /* save index to element for later sort */
732 4          finx(i) = i;
733 4          do kb = 0 to fcbn;
734 5              fcbv(kb) = bfcbl(kb);
735 5          end;
736 4          fcb$e,fcb$k,fcb$r(i) = 0;
737 4          end copy$fcbl;

```

CP/M-86 v.1.1 (vol. II)

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # C81-162

```
/* ----- */
```

```

/* update statistics with current fcb blocks */
738 3  add$fcbs$blks: proc;
739 4  if bfc(0) < 10h then do;
      /* NOT AN XFCB */

      /* set any attribute that is on */
741 5  do kb=1 to fnam;
742 6  if (bfc(kb) and 80h) = 80h then
743 6  fcbv(kb) = fcbv(kb) or 80h; /* set */
744 6  end;

      /* reset archived attribute if it is off */
745 5  if (bfc(archiv) and 80h) <> 80h then
746 5  fcbv(archiv) = fcbv(archiv) and 7fh; /* reset */

      fcb = fcb + 1; /* extent incremented */
747 5  nfcbs = nfcbs + 1;

      /* record count */
749 5  fcb(i) = fcb(i) + bfc(frc)
      + (bfc(fext) and dph$byte(extask$b)) * 128;
      /* count kilobytes */
750 5  call count$blks(false);
751 5  end;
752 4  else if bfc(0) < 20h then do;
      /* XFCB */
754 5  xfc = xfc or 80h; /* bit 7 = xfc exists flag */
755 5  nxfcbs = nxfcbs + 1;
756 5  end;
      end add$fcbs$blks;

/* ----- */
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 53
PACIFIC GROVE, CA 93950
SER. # _____

```

/* sort the file names in ascending order */
758 3  sort: procedure;
759 4  if fcbn > 1 then /* requires at least two to sort */
760 4  do; l = 1;
762 5  do while l > 0; /* bubble sort */
763 6  call test$break;
764 6  l = 0;
765 6  do m = 0 to fcbn - 2;
766 7  i = finx(m+1); call multi16; bfcba = fcbba; i = finx(m);
770 7  call multi16; /* sets fcbba, basing fcbv */
771 7  do kb = 1 to fnam; /* compare for less or equal */
772 8  if (b:=(bfc(kb) and 7fh))
      < (f:=(fcbv(kb) and 7fh))
      then /* switch */
773 8  do; k = finx(m); finx(m) = finx(m + 1);
776 9  finx(m + 1) = k; l = l + 1; kb = fnam;
779 9  end;
780 8  else if b > f then kb = fnam; /* stop compare */
      end;
783 7  end;
784 6  end;
end;
```

```

785 5      end;
786 4      end sort;
/* ----- */

/* display the fcbs */
787 3      display: proc;
788 4          dcl (wide, add, sizecols) byte;

789 4          add, sizecols = 0;
790 4          if (wide := (columns > 48)) then
791 4              add = 7;
792 4          if sizeset then
793 4              add = add + (sizecols := 10);
794 4          call print(.drivename);
795 4          call show$drive;
796 4          call blanks(17+add);
797 4          call show$usr(user$code);
798 4          if sizeset then
799 4              call print(('      Size ',0));
800 4          else
801 4              call crlf;
802 4          call printx((' Recs  Bytes FCBs Attrib',0));
803 4          if wide then
804 4              call printx(('      utes  ',0));
805 4          call printx(('      Name',0));
806 4          l = 0;
807 5          do while l < fcbr;
808 5              i = finx(l); /* i is the index to next in order */
/* print the file length */
810 5              call move(16,.fcbv(0),fcba);
811 5              fcb(0) = 0;

/* display size */

812 5          if sizeset then
813 5              do;
814 6                  call getfilesize(fcba);
815 6                  call p3byte(rreca,true);
816 6                  call printb;
817 6              end;

/* display records */

818 5          call pdecimal(fcbr(i),10000); /* rrrrr */
819 5          call printb;

/* display kbytes increment 1-kblocks (kblks) */

820 5          kblks = (fcbr(i) / 8) + kblks;
821 5          if fcbr(i) mod 8 <> 0 then
822 5              kblks = kblks + 1;
823 5          call pdecimal(fcbk,10000); /* bbbbbb */
824 5          tall = tall + fcbk;
825 5          call printchar('K'); call printb;

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER # _____

```

/* is there an xfcbs ? (check hi-bit of s1 byte) */

827 5      xfcbsfound = attribute(fsl);      /* save for xfcbs column */
828 5      xfcbs = xfcbs and 7fh;      /* clear bit7 */

/* display # fcbs */

829 5      call pdecimal(fcbs,1000);      /* eeee */

830 5      call printb;

/* display attributes: sys,ro,x,a,f1-f4 */

831 5      if attribute(infile) then
832 5          call printx(('.Sys',0));
            else
833 5          call printx(('.Dir',0));
834 5          call printb;
835 5          call printchar('R');
836 5          if attribute(rofile) then
837 5              call printchar('0');
            else
838 5              call printchar('W');
839 5          call printb;
840 5          if wide then do;
            /* display # xfcbs (0 or 1) */
842 6          if xfcbsfound then
843 6              call printchar('X');
            else
844 6              call printb;
845 6              call prnt$attrib(archiv,'A');
846 6              call prnt$attrib(attrb1,'1');
847 6              call prnt$attrib(attrb2,'2');
848 6              call prnt$attrib(attrb3,'3');
849 6              call prnt$attrib(attrb4,'4');
850 6              call printb;
851 6              end;

/* display filename */

/* print filename.typ */
852 5      call printfn(fcbsa);

/* increment fcbs counter */

853 5      l = l + 1;
854 5      end;

/* skip past size and recs fields */
855 4      call dots(39+add);
856 4      call print(('.Total:',0));
857 4      call blanks(sizecols);
858 4      call pdecimal(tall,10000);      /* total kbytes */
859 4      call printchar('K');
860 4      call pdecimal(nfcbs,10000);      /* total # fcbs */

/*      call blanks(2);
      call printchar('(');
      call pvalue(fcbn);

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. Box 579

PACIFIC GROVE, CA 93950

SER. # _____

```

      call printx(,(' file',0));
      if fcbn < 2 then
        call printx(,(' ',0));
      else
        call printx(,('s, ',0));
      call pvalue(kblks);
      call printx(,(' -1k blocks',0));
      call printchar(' '); */
end display;
861 4
/* ----- */

```

```

      /* set file attributes */
862 3 setfileatt: proc;
863 4   l = 0;
864 4   do while l < fcbn;
865 5     i = 1;
866 5     call multil6;
867 5     call crlf;
868 5     call printfn(fcb$a);
869 5     call printx(,set$t0);
870 5     call set$attributes(scascl-1);
871 5     if scas2 <> 0 then do;
873 6       call printx(,(' ',0));
874 6       call set$attributes(scas2-1);
875 6     end;
876 5     fcbv(0) = 0; /* incase matched user # > 0 */
877 5     call setind(fcb$a);
878 5     l = l + 1;
879 5     end;
880 4   end setfileatt;
/* ----- */

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950
 SER. # _____

```

      /* MAIN ROUTINE */
881 3 if (set$attribute := setfilestatus) then
882 3   if scascl = 0 then
883 3     return;

/* read the directory, collect all common file names */
/* skip allocation vector at the base of free memory
   used by the duplicate block checking routines */

884 3 membase = (i := shr(dpb$word(blkmax$w)+7,3)) + .memory;
885 3 call fill(.memory,0,i); /* clear allocation vector */
886 3 lb = 1;
887 3 if (word$blks := (dpb$word(blkmax$w) > 255)) then
888 3   lb = 2; /* double precision block indices */
889 3 fcbn,fcb(0) = 0;
890 3 fcb(fext),fcb(fmod) = '?'; /* question mark matches all */
891 3 call search(.ufcb); /* fill directory buffer */
892 3 call check$user;

/* collect fcb's for display */
893 3 collect: /* label for debug */

do while dcnt < 255;

```

```

/* another item found, compare it for common entry */
894 4      bfcba = shl(dcnt and 11b,5)+buffa; /* dcnt mod 4 * 32 */
895 4      call compare$fcba;
896 4      if matched then
897 4          i = i - 1;
          else
898 4          call copy$fcba;
/* entry is at, or was placed at location i in fcbs
~:bsa = finx(i) */
899 4      call add$fcba$blks;
900 4      call searchn; /* to next entry in directory */
901 4      call check$user;
902 4      end; /* of do while dcnt <> 255 */

903 3      if n~t first$error then
904 3          call boot; /* abort if duplicate block error */
905 3      if fcba = 0 then
906 3          call print(.'File Not Found',0));
907 3      else if not set$attribute then
          /* display collected data */
908 3          do;
909 3          call sort;
910 3          call display;
911 3          call pralloc;
912 3          end;
          else
913 3          call setfileatt;
914 3          end getfile;
/* ----- */

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 578
 PACIFIC GROVE, CA 93950

SER. = _____

```

/* reset drive (d:=RW) */
715 2      reset$drv: procedure byte;
716 3          dcl b address;

717 3          b = 1;
718 3          if cdisk <> 0 then
719 3              b = shl(double(1),cdisk);
720 3          return mon2(37,b);
721 3          end reset$drv;
/* ----- */

```

```

/* check for drive status assignment */
922 2      setdrivestatus: procedure;
923 3          declare i byte;

924 3          prdrive: proc(a);
925 4              dcl a addr;
926 4              call print(.drivename(1));
927 4              call show$drv;
928 4              call printx(.set$to);
929 4              call printx(a);
930 4              end prdrive;

931 3          if (i := match(.attrib1,2)) = 1 then do;
933 4              call writeprot;

```

```

934 4      call prdrive(.readonly);
935 4      end;
936 3      else if i = 2 then do;
938 4          if reset$drv < 0 then
939 4              call print(.'Disk Reset Denied',0));
          else
940 4              call prdrive(.readwrite);
941 4              end;
942 3      else do;
943 4          call print(.invalid);
944 4          call print(.use);
945 4          call printx(.'d:=RW',0));
946 4          end;
947 3      end setdrivestatus;
/* ----- */

```

```

/* reset drive (d:=RW) */
948 2      parsetit: procedure;
949 3          dcl i byte;

950 3          if (i:=match(.dev1,8)) = 8 then
951 3              call drivestatus;
952 3          else if i = 7 then
953 3              call userstatus;
          else
954 3              call getfile;
955 3          end parsetit;
/* ----- */

```

```

/* process request (main procedure body) */

956 2      ver = version;
957 2      cdisk = cselect;
958 2      user$code = getuser;
959 2      bdos3 = (high(ver) = bdos3ver);
          /* size display if $S set in command */
960 2      if not parset$next then
961 2          call prstatus;
962 2      else if accum(1) = ':' then do;
964 3          call select$disk(accum(0)-'A');
965 3          if not parset$next then
966 3              call pralloc;
          else if accum(0) = '=' then do;
967 3              call scan;
968 4              call setdrivestatus;
969 4              call set$kp;
970 4              call getfile;
971 4              end;
          else
972 3              call parsetit;
973 3          end;
974 2      else do;
975 3          call set$kp;
976 3          if not devreq then
977 3              call getfile;
978 3          end;

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

979 2 return;
980 2 end pla;
981 1 end;

COPYR GHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. # _____

CROSS-REFERENCE LISTING

DEFN ADDR SIZE NAME, ATTRIBUTES, AND REFERENCES

170	0000H	1	A.	BYTE BASED(S)	172					
65	0004H	2	A.	WORD PARAMETER AUTOMATIC		66	67	68	69	70
268	0000H	7	A.	BYTE BASED(AP) ARRAY(7)	271					
96	0004H	2	A.	WORD PARAMETER AUTOMATIC	97	99				
623	0004H	2	A.	WORD PARAMETER AUTOMATIC	624	626	627			
570	0004H	2	A.	WORD PARAMETER AUTOMATIC	571	576				
924	0004H	2	A.	WORD PARAMETER AUTOMATIC	925	929				
138	0004H	2	A.	WORD PARAMETER AUTOMATIC	139	140				
435	0004H	2	A.	WORD PARAMETER AUTOMATIC	436	437	438	439		
42	0853H	4	ACCUM.	BYTE ARRAY(4)	181	190	204	211	214	423 679 962 964 967
788	03D8H	1	ADD.	BYTE	789	791	793	796	855	
267	04BAH	75	ADD.	PROCEDURE STACK=0006H	284	287				
315	0618H	15	ADDBLOCK	PROCEDURE STACK=0004H	326					
738	1077H	166	ADDFCBLKS	PROCEDURE STACK=002EH	899					
3			ADDR	LITERALLY	170	925				
315	0004H	2	AK	WORD PARAMETER AUTOMATIC	316	317	318			
40	0006H	2	ALLBLKS.	WORD	323	324	325			
632	0004H	1	ALLOC.	BYTE PARAMETER AUTOMATIC	633	642				
591	0D10H	169	ALLOCATE	PROCEDURE BYTE STACK=0024H	643					
592	08D2H	1	AMASK.	BYTE	602	604	605	620		
267	0006H	2	AP	WORD PARAMETER AUTOMATIC	268	271				
559			ARCHIV	LITERALLY	745	746	845			
559			ATTRB1	LITERALLY	846					
559			ATTRB2	LITERALLY	847					
559			ATTRB3	LITERALLY	848					
559			ATTRB4	LITERALLY	849					
43	00D3H	20	ATTRIBL.	BYTE ARRAY(20) DATA	681	686	931			
579	0CD4H	32	ATTRIBUTE.	PROCEDURE BYTE STACK=0004H	587	827	831	836		
43	007FH	25	ATTRIBUTES	BYTE ARRAY(25) DATA	461	675				
59	0004H	1	B.	BYTE PARAMETER AUTOMATIC	60	61				
178	0004H	1	B.	BYTE PARAMETER AUTOMATIC	179	181				
202	089DH	1	B.	BYTE						
585	0006H	1	B.	BYTE PARAMETER AUTOMATIC	586	587				
177	089CH	1	B.	BYTE	184	187	191	192	195	196 198
268	0000H	7	B.	BYTE BASED(BP) ARRAY(7)	271	272	273			
561	08C7H	1	B.	BYTE	772	780				
916	0846H	2	B.	WORD	917	919	920			
298	0000H	3	B3	STRUCTURE BASED(BYTE3ADR)	299	300				
331	0000H	3	B3	STRUCTURE BASED(B3A)	340	341	348	349		
291	0000H	3	B3	STRUCTURE BASED(BYTE3ADR)	292	293	294			
277	0000H	3	B3	STRUCTURE BASED(BYTE3ADR)	280	281				
331	002AH	2	B3A.	WORD	331	334	340	341	344	348 349 350 354
16	0000H		BASEDPB.	PROCEDURE EXTERNAL(3) STACK=0000H	159					
234	03F3H	76	BCD.	PROCEDURE STACK=0004H	280					
40	084EH	1	BDO33.	BYTE	515	959				
2			BDO3VER	LITERALLY	959					
39	0000H	32	BFCB	BYTE BASED(BFCBA) ARRAY(32)	745	749	752	772	627	637 639 643 734 739 742
39	0000H	2	BFCBA.	WORD	39	615	617	627	637	639 643 653 654 658 734 739

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #

			742	745	749	752	768	772	894	
39	0000H	1	BFCBUSER	BYTE BASED(BFCBA)				615	654	658
59	00A1H	24	BLANKS	PROCEDURE STACK=0014H					616	796 857
24			BLKMAXW.	LITERALLY	323	324	349	884	887	
24			BLKMSKB.	LITERALLY	299					
24			BLKSHFB.	LITERALLY	160	162	299	300	366	
592	0000H	1	BLOCKBYTE.	BYTE BASED(BLOCKPTR)				599	600	
591	0004H	2	BLOCKPTR	WORD PARAMETER AUTOMATIC				592	595	596 599 600
592	0000H	2	BLOCKWORD.	WORD BASED(BLOCKPTR)				595	596	
3			BOOLEAN.	LITERALLY						
49	0075H	14	BOOT	PROCEDURE STACK=0008H				88	730	904
267	0004H	2	BP	WORD PARAMETER AUTOMATIC				268	271	272 273
80	00F4H	15	BREAK.	PROCEDURE BYTE STACK=0008H					84	
38	0000H	128	BUFF	BYTE ARRAY(128) EXTERNAL(17)					184	191 193 653 894
38			BUFFA.	LITERALLY	653	894				
41	0850H	3	BYTE3.	STRUCTURE	344	517	519	522	523	525
290	0004H	2	BYTE3ADR	WORD PARAMETER AUTOMATIC				291	292	293 294 295
276	0006H	2	BYTE3ADR	WORD PARAMETER AUTOMATIC				277	280	281
297	0004H	2	BYTE3ADR	WORD PARAMETER AUTOMATIC				298	299	300
268	08A3H	1	C.	BYTE	269	271	272			
585	0004H	1	C.	BYTE PARAMETER AUTOMATIC				586	588	
246	0004H	1	C.	BYTE PARAMETER AUTOMATIC				247	256	
169	0004H	2	C.	WORD PARAMETER AUTOMATIC				170	171	
40	084FH	1	CDISK.	BYTE	110	303	518	533	534	918 919 957
52	0004H	1	CHAR	BYTE PARAMETER AUTOMATIC				53	54	
648	0E6DH	80	CHECKUSER.	PROCEDURE STACK=002EH				892	901	
24			CHKSIZW.	LITERALLY	360					
30	0000H	1	CKDRV.	BYTE EXTERNAL(8)						
893	0C13H		COLLECT.	LABEL						
28	0000H		COLUMNS.	PROCEDURE BYTE EXTERNAL(7) STACK=0000H						790
715	0FC4H	59	COMPAREFCB	PROCEDURE STACK=0008H				895		
724	0FFFH	120	COPYFCB.	PROCEDURE STACK=0022H				898		
320	062AH	75	COUNT.	PROCEDURE WORD STACK=0008H					523	
632	0DF5H	120	COUNIBLKS.	PROCEDURE STACK=002AH				656	750	
3			CR	LITERALLY	87	92				
91	0123H	20	CRLF	PROCEDURE STACK=0018H				98	333	374 379 383 400 452 457
					466	474	530	556	565	614 800 809 867
125	01B9H	15	CSELECT.	PROCEDURE BYTE STACK=0008H					957	
403	0889H	1	D.	BYTE	405	409	413			
107	0004H	1	D.	BYTE PARAMETER AUTOMATIC				108	110	111
153	0004H	1	D.	BYTE PARAMETER AUTOMATIC				154	155	
164	0004H	1	D.	BYTE PARAMETER AUTOMATIC				165	166	
220	089FH	1	D.	BYTE	223	226	230			
544	08C3H	1	D.	BYTE	546	550	554			
3			DCL.	LITERALLY						
39	0848H	1	DCNT	BYTE	115	120	123	391	392	651 653 893 894
43	0088H	32	DEVL	BYTE ARRAY(32) DATA				447	467	478 950
433	00ECH	64	DEVR	BYTE ARRAY(64) DATA				449	471	497
432	08E4H	214	DEVREQ	PROCEDURE BYTE STACK=0036H					976	
443	09D9H	98	DEVSTATUS.	PROCEDURE STACK=001CH				486		
24			DIRBLKW.	LITERALLY						
43	0034H	16	DIRECTORY.	BYTE ARRAY(16) DATA				707		
24			DIRMAXW.	LITERALLY	357					
402	0848H	50	DISKSTATUS	PROCEDURE STACK=0032H				492		
787	11E7H	529	DISPLAY.	PROCEDURE STACK=0024H				910		
128	0004H	2	DMA.	WORD PARAMETER AUTOMATIC				129	130	
38	0011H	1	DOLL	BYTE EXTERNAL(9) AT						

COPYRIGHT © 1981

BY MICROSOFT CORP.

ALL RIGHTS RESERVED

38		DOLLA.	LITERALLY	38																
563	0C7BH	30 DOTS	PROCEDURE STACK=001EH		855															
		DOUBLE	BUILIN	162 366 919																
21	0000H	DPBYTE.	PROCEDURE BYTE EXTERNAL(5) STACK=0000H			160	162	299	300	363										
			366 749																	
21	0000H	1 DPINDEX	BYTE PARAMETER	22																
18	0000H	1 DPINDEX	BYTE PARAMETER	19																
18	0000H	DPWORD.	PROCEDURE WORD EXTERNAL(4) STACK=0000H			323	324	349	357	360										
			369 372 884 887																	
43	0000H	8 DRIVENAME.	BYTE ARRAY(8) DATA	794 926																
330	0675H	248 DRIVESTATUS.	PROCEDURE STACK=002EH		410 951															
25	0000H	2 DSH.	WORD PARAMETER	26																
43	0044H	20 ENTRIES.	BYTE ARRAY(20) DATA	359 362																
510	09AFH	ERROR.	LABEL	481 496 498																
670	0F3EH	40 ERROR.	PROCEDURE STACK=0022H		684 687															
24		EXTHSKB.	LITERALLY	363 749																
169	0006H	1 F.	BYTE PARAMETER AUTOMATIC		170 172															
561	08C3H	1 F.	BYTE	772 780																
44	0866H	7 F0	BYTE ARRAY(7) INITIAL		44															
44	086DH	7 F1	BYTE ARRAY(7) INITIAL		44															
44	0874H	7 F2	BYTE ARRAY(7) INITIAL		44															
44	087BH	7 F3	BYTE ARRAY(7) INITIAL		44															
44	0882H	7 F4	BYTE ARRAY(7) INITIAL		44															
44	0889H	7 F5	BYTE ARRAY(7) INITIAL		44															
44	0890H	7 F6	BYTE ARRAY(7) INITIAL		44															
44	089FH	7 FAC.	BYTE ARRAY(7) INITIAL		279 284 287															
3		FALSE.	LITERALLY	205 215 229	257 387 424 482 583 609 618 628															
			678 716 750																	
117	0004H	2 FCB.	WORD PARAMETER AUTOMATIC		118 119 120															
149	0004H	2 FCB.	WORD PARAMETER AUTOMATIC		150 151															
31	0000H	1 FCB.	BYTE ARRAY(1) EXTERNAL(9)		38 657 810 811 814 815 889 890															
624	0000H	16 FCB.	BYTE BASED(A) ARRAY(16)		626 627															
113	0004H	2 FCB.	WORD PARAMETER AUTOMATIC		114 115															
119	0000H	1 FC80	BYTE BASED(FC8)																	
32	0000H	1 FCB16.	BYTE ARRAY(1) EXTERNAL(10)																	
38		FCBA	LITERALLY	810 814																
560	0000H	2 FCBE	WORD BASED(FCBEA)	736 747 829																
560	0838H	2 FCSEA.	WORD	560 665 736 747 829																
560	0000H	2 FCBK	WORD BASED(FCBMA)	644 736 823 824																
560	083CH	2 FCBHA.	WORD	560 644 666 736 823 824																
40		FCBMAX	LITERALLY	560 727																
560	0034H	2 FCBN	WORD	718 725 727 759 765 806 864 889 905																
560	0436H	1024 FCBR	WORD ARRAY(512)	736 749 818 820 821																
560	0000H	1 FCBS	BYTE BASED(MEMBASE) ARRAY(1)																	
560	083AH	2 FCBS1.	WORD	560 667 754 828																
560	0836H	2 FCDSA.	WORD	560 581 665 666 667 694 698 702 706 720 727 734																
			743 746 768 772 810 852 868 876 877																	
571	0000H	16 FCBV	BYTE BASED(A) ARRAY(16)		576															
560	0000H	16 FCBV	BYTE BASED(FCBSA) ARRAY(16)		581 694 698 702 706 734 743															
			746 772 810 876																	
559		FEXT	LITERALLY	665 749 890																
3		FF	LITERALLY																	
623	01B9H	60 FILEMATCH.	PROCEDURE BYTE STACK=0004H		657 720															
43	0058H	16 FILENAME	BYTE ARRAY(16) DATA	460 673																
169	028DH	32 FILL	PROCEDURE STACK=0008H		190 278 279 885															
360	0036H	1024 FINX	WORD ARRAY(512)	732 766 769 774 775 776 807																

CONFIDENTIAL 1981, 1992
 DIGITAL RESEARCH
 1000 BAY ST
 FARMINGDALE, N.Y. 11735

43	0373H	1	FIRSTERROR	BYTE INITIAL	607	609	903													
244	0022H	1	FIXED.	BYTE PARAMETER	245	252														
513	0004H	1	FIXED.	BYTE PARAMETER AUTOMATIC			514	525												
276	0004H	1	FIXED.	BYTE PARAMETER AUTOMATIC			277	288												
559			FMOD	LITERALLY	666	870														
559			FNAM	LITERALLY	573	733	741	771	778	781										
3			FOREVER.	LITERALLY	477	650														
559			FRC.	LITERALLY	749															
559			FSI.	LITERALLY	667	827														
559			FTYP	LITERALLY	574															
4	0000H	1	FUNC	BYTE PARAMETER	5															
12	0000H	1	FUNC	BYTE PARAMETER	13															
8	0000H	1	FUNC	BYTE PARAMETER	9															
25	0000H		GETALL	PROCEDURE WORD EXTERNAL(6)	STACK=0000H					323										
558	0893H	224	GETFILE.	PROCEDURE STACK=0032H		954	977													
147	0228H	16	GETFILESIZE. . . .	PROCEDURE STACK=000AH		814														
153	0238H	19	GETFREESP.	PROCEDURE STACK=000AH		518														
132	01D8H	15	GETLOGIN	PROCEDURE WORD STACK=0008H			404	545												
104	0156H	15	GETRODISK.	PROCEDURE WORD STACK=0008H		109														
142	0206H	15	GETUSER.	PROCEDURE BYTE STACK=0008H		958														
331	0002H	1	HBYTE.	BYTE MEMBER(B3)	340	348														
41	0002H	1	HBYTE.	BYTE MEMBER(BYTE3)		522														
298	0002H	1	HBYTE.	BYTE MEMBER(B3)	299															
291	0002H	1	HBYTE.	BYTE MEMBER(B3)	292	293														
277	0002H	1	HBYTE.	BYTE MEMBER(B3)	281															
			HIGH	BUILTIN	959															
277	08A6H	1	HIGHBYTE	BYTE	281	283	285													
245	08A2H	1	I.	BYTE	248	250	260	261	262	263										
235	08A0H	1	I.	BYTE	237	239														
949	08DBH	1	I.	BYTE	950	952														
177	089BH	1	I.	BYTE	180	181	182	189												
321	0026H	2	I.	WORD	325															
563	0004H	1	I.	BYTE PARAMETER AUTOMATIC			564	566												
633	08D4H	1	I.	BYTE	635	636	637	639	643											
624	08D3H	1	I.	BYTE	625	626	627													
579	0004H	1	I.	BYTE PARAMETER AUTOMATIC			580	581												
923	08DAH	1	I.	BYTE	931	936														
561	083EH	2	I.	WORD	665	717	718	721	725	732	736	749	766	769	807	818				
					820	821	865	884	885	897										
418	08BAH	1	I.	BYTE	422	423														
434	08BEH	1	I.	BYTE	446	447	465	467	471	478	485	487	489	491	494	500				
649	08D6H	1	I.	BYTE																
377	08ABH	1	I.	BYTE	386	387	392	393	396	397	398									
277	03A5H	1	I.	BYTE	282	284														
268	06A4H	1	I.	BYTE	270	271	272	273												
42	0357H	1	IBP.	BYTE INITIAL	184	185	188	191	193	194	199									
559			INFILE	LITERALLY	702	706	831													
12	0000H	2	INFO	WORD PARAMETER	14															
8	0000H	2	INFO	WORD PARAMETER	10															
4	0000H	2	INFO	WORD PARAMETER	6															
43	0073H	19	INVALID.	BYTE ARRAY(19) DATA		510	671	710	943											
434	08C0H	1	IOBYTE	BYTE	444	449	451	499	501	504										
434	08C2H	1	IOVAL.	BYTE	504	505														
40	034AH	1	IOVAL.	BYTE																
73	00D2H	15	IOVALF	PROCEDURE BYTE STACK=0008H			444	504												
434	08C1H	1	ITEMS.	BYTE	476	480	484													

COPYRIGHT 1981, 1982

DIGITAL RESEARCH

P.O. BOX 579

PALM GROVE, CA 93950

SER. ==

434	03BFH	1	J.	BYTE	445	449	450	469	471	494	497	502	504
418	08BRH	1	J.	BYTE	419	423	425						
571	08CEH	1	K.	BYTE	573	574	576						
561	0842H	2	K.	WORD	774	776							
321	0024H	2	KA	WORD	322	326	328						
317	0000H	2	KACCUH	WORD BASED(AK)			318						
561	08C4H	1	KB	BYTE	733	734	741	742	743	771	772	778	781
46	0018H	2	KBLKS.	WORD INITIAL			820	822					
40	0004H	1	KPB.	WORD	161	162	318	644					
158	089AH	1	KSHF	BYTE	160								
561	0840H	2	L.	WORD	761	762	764	777	805	806	807	853	863
												864	865
												878	
			LAST	BUILTIN		386	396						
47	0899H	1	LASTDSEGBYTE . . .	BYTE INITIAL									
571	08CFH	1	LB	BYTE									
561	08CSH	1	LB	BYTE	636	886	888						
45	0897H	1	LB	BYTE INITIAL									
34	0000H	1	LENO	BYTE EXTERNAL(12)									
36	0000H	1	LEN1	BYTE EXTERNAL(14)									
3			LF	LITERALLY		87	93						
3			LIT.	LITERALLY		3	24						
543	0030H	2	LOGIN.	WORD	545	547	548	553					
403	002CH	2	LOGIN.	WORD	404	406	407	412					
			LOW.	BUILTIN		407	536	548					
331	0000H	2	LWORD.	WORD MEMBER(B3)			341	349					
41	0000H	2	LWORD.	WORD MEMBER(BYTE3)				523					
298	0000H	2	LWORD.	WORD MEMBER(B3)			300						
291	0000H	2	LWORD.	WORD MEMBER(B3)			294						
277	0000H	2	LWORD.	WORD MEMBER(B3)			280						
561	0844H	2	M.	WORD	765	766	769	774	775	776			
418	08BCH	1	MATCH.	BYTE	421	424	427						
416	037AH	106	MATCH.	PROCEDURE BYTE STACK=0006H					478	497	681	666	931
561	08CBH	1	MATCHED.	BYTE	716	718	720	896					
321	0023H	2	MAXALL	WORD	324	325							
38	0000H	2	MAXB	WORD EXTERNAL(16)				727					
561	08C6H	1	MB	BYTE	637	639	640						
560	0032H	2	MEMBASE.	WORD	560	665	884						
	0000H		MEMORY	BYTE ARRAY(0)		389	392	605	620	884	885		
38			MEMSIZE.	LITERALLY		727							
4	0000H		MON1	PROCEDURE EXTERNAL(0) STACK=0000H					50	54	78	86	111
						136	140	147	151	155			
8	0000H		MON2	PROCEDURE BYTE EXTERNAL(1) STACK=0000H					74	81	115	120	123
						126	143	920					
12	0000H		MON3	PROCEDURE WORD EXTERNAL(2) STACK=0000H					102	105	133		
			MOVE	BUILTIN		810							
664	0EBDH	43	MULTI16.	PROCEDURE STACK=0002H				719	726	767	770	808	866
46	001AH	2	NFCBS.	WORD INITIAL			748	860					
633	08D5H	1	NOERROR.	BYTE	634	636	643						
46	001CH	2	NXFCBS	WORD INITIAL			755						
24			OFFSETW.	LITERALLY		372							
113	0181H	19	OPEN	PROCEDURE STACK=000AH									
276	0505H	117	P3BYTE	PROCEDURE STACK=0020H				334	525	815			
38	0011H	1	PARM	BYTE EXTERNAL(9) AT									
38			PARMA.	LITERALLY		38							
201	0360H	24	PARSEASSIGN. . . .	PROCEDURE BYTE STACK=0010H				495					
948	14EFH	42	PARSEIT.	PROCEDURE STACK=0036H				972					
209	0378H	31	PARSENEXT.	PROCEDURE BYTE STACK=0010H					507	677	685	960	965

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #

33	0000H	2	PASS0.	WORD EXTERNAL(11)															
35	0000H	2	PASS1.	WORD EXTERNAL(13)															
246	0484H	54	PCHAR.	PROCEDURE STACK=0014H	262	264													
219	0397H	92	PDECIMAL	PROCEDURE STACK=0016H	313	398	818	823	829	858	860								
48	0000H	117	PLM.	PROCEDURE PUBLIC STACK=003AH															
528	0B1FH	71	PRALLOC.	PROCEDURE STACK=002AH	551	911	966												
513	0ADFH	64	PRCOUNT.	PROCEDURE STACK=0026H	540														
924	14D1H	30	PRDRIVE.	PROCEDURE STACK=0024H	934	940													
235	0020H	2	PREC	WORD 236 238 239 240	241														
219	0004H	2	PREC	WORD PARAMETER AUTOMATIC	220	222	223	224	225	226									
96	0137H	16	PRINT.	PROCEDURE STACK=001EH	345	458	459	462	463	464	510	610							
				612 671 672 710 729 794	799	856	906	926	939	943	944								
56	0096H	11	PRINTB	PROCEDURE STACK=000EH	62	227	253	308	336	589	816	819							
				826 830 834 839 844 850															
244	043FH	69	PRINTBCD	PROCEDURE STACK=0018H	288														
52	0033H	19	PRINTCHAR.	PROCEDURE STACK=000AH	57	69	92	93	230	256	303	304							
				335 438 441 470 526 532	537	538	567	575	576	588	825	835							
				837 838 843 859															
570	0C99H	59	PRINTFN.	PROCEDURE STACK=0014H	617	852	868												
65	0089H	25	PRINTX	PROCEDURE STACK=0010H	87	99	312	347	352	353	356	358							
				359 361 362 364 365 367	368	370	371	373	381	385	448	456							
				460 461 468 539 673 674	675	695	699	703	707	801	803	804							
				832 833 869 873 928 929	945														
435	09BAH	31	PRNAME	PROCEDURE STACK=0010H	447	449	467	471											
585	0CF4H	28	PRNTATTRIB	PROCEDURE STACK=0016H	845	846	847	848	849										
3			PROC	LITERALLY 169 443 715	724	738	787	862	924										
542	0B66H	53	PRSTATUS	PROCEDURE STACK=002EH	961														
44	000AH	14	PTR.	WORD ARRAY(7) INITIAL	234														
338	078BH	23	PV	PROCEDURE STACK=002AH	357	360	363	366	369	372									
332	076DH	27	PV3.	PROCEDURE STACK=0024H	342	351	355												
43	0008H	15	READONLY	BYTE ARRAY(15) DATA	695	934													
43	0017H	16	READWRITE.	BYTE ARRAY(16) DATA	699	940													
43	008FH	16	RECORDMSG.	BYTE ARRAY(16) DATA	352	364	367	370											
915	1456H	40	RESETDRV	PROCEDURE BYTE STACK=0008H	938														
529	002EH	2	RO	WORD 534 535 536															
40	0008H	2	RODISK	WORD 109 534 535															
559			ROFILE	LITERALLY 694 698 836															
			ROL.	BUILTIN 299 501 581															
			ROR.	BUILTIN 292															
38	0023H	1	ROVF	BYTE EXTERNAL(9) AT															
38	0021H	2	RREC	WORD EXTERNAL(9) AT															
38			RRECA.	LITERALLY 38 815															
38			RRECD.	LITERALLY 38															
169	0008H	2	S.	WORD PARAMETER AUTOMATIC	170	172	173												
67	0000H	1	S.	BYTE BASED(A) 68 69															
176	02A0H	148	SCAN	PROCEDURE STACK=000CH	203	206	210	213	680	969									
690	0004H	1	SCASE.	BYTE PARAMETER AUTOMATIC	691	692													
562	08CCH	1	SCASE1	BYTE 681 683 870 882															
562	08CDH	1	SCASE2	BYTE 686 871 874															
117	0194H	19	SEARCH	PROCEDURE STACK=000AH	390	891													
122	01A7H	18	SEARCHN.	PROCEDURE STACK=0008H	394	661	900												
3			SECTORLEN.	LITERALLY															
107	0165H	28	SELECT	PROCEDURE STACK=000EH	166														
164	027DH	16	SELEC/DISK	PROCEDURE STACK=0014H	409	550	964												
178	0341H	31	SETACC	PROCEDURE STACK=0004H	196														
40	084CH	1	SETATTRIBUTE	BYTE INITIAL 881 907															
690	0F66H	94	SETATTRIBUTES.	PROCEDURE STACK=0024H	870	874													

COPYRIGHT - 1981, 1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER --

128	01C8H	16	SETDMA	PROCEDURE STACK=000AH	389	517														
922	147EH	83	SETDRIVESTATUS . .	PROCEDURE STACK=0028H	970															
862	13FAH	92	SETFILEATT	PROCEDURE STACK=0028H	913															
669	0EE8H	36	SETFILESTATUS. . .	PROCEDURE BYTE STACK=0026H		881														
138	01F6H	16	SETIND	PROCEDURE STACK=000AH	877															
157	024BH	50	SETKPB	PROCEDURE STACK=0006H	167	975														
43	0086H	9	SETTD.	BYTE ARRAY(9) DATA	869	928														
145	0215H	19	SETUSER.	PROCEDURE STACK=000AH																
			SHL.	BUILTIN	2	299	300	360	366	392	447	449	467	471	494					
					502	653	665	894	919											
297	0598H	73	SHL3BYTE	PROCEDURE STACK=000AH	350															
306	03F5H	11	SHOWDRIVE.	PROCEDURE STACK=0012H	346	380	384	531	795											
302	05E1H	20	SHOWDRV.	PROCEDURE STACK=000EH	307	572	611	927												
310	0600H	27	SHOWUSR.	PROCEDURE STACK=001CH	382	615	797													
			SHR.	BUILTIN	275	293	294	412	451	534	553	595	599	604	884					
290	057AH	30	SHR3BYTE	PROCEDURE STACK=0004H	354	519														
76	00E1H	19	SIOVALF.	PROCEDURE STACK=000AH	505															
788	08D9H	1	SIZECOLS	BYTE	789	793	857													
40	084BH	1	SIZESET.	BYTE INITIAL	682	792	798	812												
758	111DH	204	SORT	PROCEDURE STACK=0018H	909															
40			SPKSHF	LITERALLY	160	162														
24			SPTW	LITERALLY	369															
1	0000H		STAT	PROCEDURE STACK=0000H																
418	08BDH	1	SYNC	BYTE	419	420	428													
43	0027H	13	SYSTEM	BYTE ARRAY(13) DATA	703															
3			TAB.	LITERALLY																
46	001EH	2	TALL	WORD INITIAL	824	858														
37	0000H	1	TBUFF.	BYTE ARRAY(1) EXTERNAL(15)																
298	0000H	2	TEMP1.	BYTE BASED(BYTE3ADR) ARRAY(2)		299														
291	0000H	2	TEMP1.	BYTE BASED(BYTE3ADR) ARRAY(2)		295														
291	08A7H	1	TEMP2.	BYTE	292	295														
83	0103H	32	TESTBREAK.	PROCEDURE STACK=0014H		94	763													
3			TRUE	LITERALLY	45	207	217	221	259	334	393	421	477	508	511					
					540	582	621	630	634	650	656	682	688	815						
39	0849H	1	UFCB	BYTE INITIAL	390	891														
43	0068H	11	USE.	BYTE ARRAY(11) DATA	672	944														
145	0004H	1	USER	BYTE PARAMETER AUTOMATIC	146	147														
378	08A7H	16	USER	BYTE ARRAY(16)	386	387	393	396	397											
310	0004H	1	USER	BYTE PARAMETER AUTOMATIC	311	313														
40	084DH	1	USERCODE	BYTE	382	658	797	958												
376	079FH	169	USERSTATUS	PROCEDURE STACK=0020H	490	953														
219	0006H	2	V.	WORD PARAMETER AUTOMATIC	220	223	224													
417	0000H	16	V.	BYTE BASED(VA) ARRAY(16)	423															
338	0004H	2	V.	WORD PARAMETER AUTOMATIC	339	341														
416	0006H	2	VA	WORD PARAMETER AUTOMATIC	417	423														
44	0358H	7	VNL.	BYTE ARRAY(7) INITIAL	239	248	262	278	287											
234	0004H	2	VALUE.	WORD PARAMETER AUTOMATIC	235	239	240													
76	0004H	1	VALUE.	BYTE PARAMETER AUTOMATIC	77	78														
455	0A3BH	164	VALUES	PROCEDURE STACK=0022H	488															
592	08D1H	1	VBIT	BYTE	596	600	603	604												
592	08D0H	1	VBYTE.	BYTE	595	599	605	620												
40	0002H	2	VER.	WORD	956	959														
101	0147H	15	VERSION.	PROCEDURE WORD STACK=0008H		956														
416	0004H	1	VL	BYTE PARAMETER AUTOMATIC	417	420														
788	08D7H	1	WIDE	BYTE	790	802	840													
561	08CAH	1	WORDBLKS	BYTE	593	638	887													
135	01E7H	15	WRITEPR01.	PROCEDURE STACK=0008H		933														

COPYRIGHT 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. #

436	0000H	1	X.	BYTE BASED(A)	437	438
560	0000H	1	XFCB	BYTE BASED(FCBS1)	754	828
561	08C9H	1	XFCBFOUND.	BYTE	827	842
245	08A1H	1	ZEROSUP.	BYTE	249	257 259
220	089EH	1	ZEROSUP.	BYTE	221	226 229

MODULE INFORMATION:

CODE AREA SIZE = 1519H 5401D
 CONSTANT AREA SIZE = 0320H 800D
 VARIABLE AREA SIZE = 08DCH 2268D
 MAXIMUM STACK SIZE = 003AH 58D
 1765 LINES READ
 0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

ISIS-II PL/M-86 V2.0 COMPILATION OF MODULE CON86

OBJECT MODULE PLACED IN CON86.OBJ

COMPILER INVOKED BY: IF0: CON86.PLM DATE(MARCH 16, 1982) XREF OPTIMIZE(3) DEBUG

```

$compact
$title ('CONSOLE 8086 - Get Console Width')
/* the purpose of this module is to allow independence */
/* of processor, i.e., 8080 or 8086 */
/* it will return a word value equal to the console width from
   the system data page. */

1      con36:
      do;

      $include (conlit.lit)
      =
2      1 = declare
      =      lit      literally      'literally',
      =      dcl      lit      'declare',
      =      proc      lit      'procedure',
      =      addr      lit      'address',
      =      true      lit      'offh',
      =      false     lit      '0',
      =      boolean   lit      'byte',
      =      forever   lit      'while true',
      =      cr        lit      '13',
      =      lf        lit      '10',
      =      tab       lit      '9',
      =      ff        lit      '12',
      =      sectorlen lit      '128';

3      1      mon4: procedure (f,a) pointer external;
4      2      dcl f byte, a address;
5      2      end mon4;

6      1      get$sysdat:
7      2      procedure pointer;
8      2      return (mon4(49,0));
9      2      end get$sysdat;

9      1      declare conwidth$pointer pointer;
10     1      declare conwidth$ptr structure (
11           offset word,
12           segment word) at (@conwidth$pointer);
11     1      declare width based conwidth$pointer byte;
12     1      declare conwidth$offset lit '0040h';

13     1      columns: procedure byte public;

14     2      conwidth$pointer = get$sysdat;
15     2      conwidth$ptr.offset = conwidth$ptr.offset + conwidth$offset;

```

COPYRIGHT © 1981-1982

BY J. R. PETERSON, JR.

0 1X 79

PACIFIC ELECTRONICS, INC. 98350

SER. # _____

```
16 2      if width = 0 then
17 2          return 80;
18 2      return width;
19 2      end columns;
```

```
20 1      end con86;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. # _____

CROSS-REFERENCE LISTING

 DEFN ADDR SIZE NAME, ATTRIBUTES, AND REFERENCES

3	0000H	2	A.	WORD PARAMETER	4				
2			ADDR	LITERALLY					
2			BOOLEAN.	LITERALLY					
13	0011H	42	COLUMNS.	PROCEDURE BYTE PUBLIC STACK=000CH					
1	0002H		CON86.	PROCEDURE STACK=0000H					
12			CONWIDTHOFFSET . .	LITERALLY	15				
9	0000H	4	CONWIDTHPOINTER. .	POINTER	10	11	14	16	18
10	0000H	4	CONWIDTHPTR. . . .	STRUCTURE AT	15				
2			CR	LITERALLY					
2			DCL.	LITERALLY					
3	0000H	1	F.	BYTE PARAMETER	4				
2			FALSE.	LITERALLY					
2			FF	LITERALLY					
2			FOREVER.	LITERALLY					
6	0002H	15	GETSYSDAT.	PROCEDURE POINTER STACK=0008H		14			
2			LF	LITERALLY					
2			LIT.	LITERALLY	2	12			
3	0000H		MON4	PROCEDURE POINTER EXTERNAL(0) STACK=0000H					7
10	0000H	2	OFFSET	WORD MEMBER(CONWIDTHPTR)	15				
2			PROC	LITERALLY					
2			SECTORLEN.	LITERALLY					
10	0002H	2	SEGMENT.	WORD MEMBER(CONWIDTHPTR)					
2			TAB.	LITERALLY					
2			TRUE	LITERALLY					
11	0000H	1	WIDTH.	BYTE BASED(CONWIDTHPOINTER)		16	18		

MODULE INFORMATION:

CODE AREA SIZE = 003BH 59D
 CONSTANT AREA SIZE = 0000H 0D
 VARIABLE AREA SIZE = 0004H 4D
 MAXIMUM STACK SIZE = 000CH 12D
 58 LINES READ
 0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

ISIS-II PL/M-86 V2.0 COMPILATION OF MODULE DPB86

OBJECT MODULE PLACED IN DPB86.OBJ

COMPILER INVOKED BY: :F0: DPB86.PLN DATE(MARCH 16, 1982) XREF OPTIMIZE(3) DEBUG

```

$compact
$title ('SDIR 8086 - Get Disk Parameters')
1  'b86:
do;
    /* the purpose of this module is to allow independence */
    /* of processor, i.e., 8080 or 8086 */

$include (comlit.lit)
=
2  1 = declare
=     lit      literally    'literally',
=     dcl      lit          'declare',
=     proc     lit          'procedure',
=     addr     lit          'address',
=     true     lit          'offh',
=     false    lit          '0',
=     boolean  lit          'byte',
=     forever  lit          'while true',
=     cr       lit          '13',
=     lf       lit          '10',
=     tab      lit          '9',
=     ff       lit          '12',
=     sectorlen lit         '128';

/* function call 32 in 2.0 or later BDOS, returns the address of the disk
parameter block for the currently selected disk, which consists of:
    spt          (2 bytes) number of sectors per track
    blkshf       (1 byte) block size = shl(double(128),blkshf)
    blkmsk       (1 byte) sector# and blkmsk = block number
    extmsk       (1 byte) logical/physical extents
    blkmax       (2 bytes) max alloc number
    dirmax       (2 bytes) size of directory-1
    dirblk       (2 bytes) reservation bits for directory
    chksiz       (2 bytes) size of checksum vector
    offset       (2 bytes) offset for operating system
*/

$include(dpb.lit)
=
= /* indices into disk parameter block, used as parameters to dpb procedure */
=
3  1 = dcl      spt$w      lit      '0',
=          blkshf$b      lit      '2',
=          blkmsk$b      lit      '3',
=          extmsk$b      lit      '4',
=          blkmax$w      lit      '5',
=          dirmax$w      lit      '7',
=          dirblk$w      lit      '9',
=          chksiz$w      lit      '11',
=          offset$w      lit      '13';
=

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. #

```
4 1 declare k$per$block byte public;
5 1 declare dpb$base pointer;
6 1 declare dpb$array based dpb$base (15) byte;

7 1 non4: procedure (f,a) pointer external;
8 2   dcl f byte, a address;
9 2 end non4;

10 1 dcl get$dpb lit '31';

11 1 dpb$byte: procedure(param) byte public;
12 2   dcl param byte;
13 2   return(dpb$array(param));
14 2 end dpb$byte;

15 1 dpb$word: procedure(param) address public;
16 2   dcl param byte;
17 2   return(dpb$array(param) + shl(double(dpb$array(param+1)),8));
18 2 end dpb$word;

19 1 base$dpb: procedure public;
20 2   dpb$base = non4(get$dpb,0);
21 2   k$per$block = shr(dpb$byte(blk$sk$b)+1,3);
22 2 end base$dpb;

23 1 end dpb86;
```

CP/M-86 v.1.1 (vol. II)

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # C81-162

CROSS-REFERENCE LISTING

DEFN ADDR SIZE NAME, ATTRIBUTES, AND REFERENCES

7	0000H	2	A.	WORD PARAMETER	8	
2			ADDR	LITERALLY		
17	003AH	38	BASEDPB.	PROCEDURE PUBLIC STACK=0008H		
3			BLKMAXW.	LITERALLY		
3			BLKXSKB.	LITERALLY	21	
3			BLKSHFB.	LITERALLY		
2			BOOLEAN.	LITERALLY		
3			CHKSIZW.	LITERALLY		
2			CR	LITERALLY		
2			DCL.	LITERALLY		
3			DIRBLKW.	LITERALLY		
3			DIRMAXW.	LITERALLY		
			DOUBLE	BUILTIN	17	
1	0002H		DPB86.	PROCEDURE STACK=0000H		
6	0000H	15	DPBARRAY.	BYTE BASED(DPBBASE) ARRAY(15)	13	17
5	0000H	4	DPBBASE.	POINTER	6	13 17 20
11	0002H	21	DPBBYTE.	PROCEDURE BYTE PUBLIC STACK=0004H		21
15	0017H	35	DPBWORD.	PROCEDURE WORD PUBLIC STACK=0004H		
3			EXTXSKB.	LITERALLY		
7	0000H	1	F.	BYTE PARAMETER	8	
2			FALSE.	LITERALLY		
2			FF	LITERALLY		
2			FOREVER.	LITERALLY		
10			GETDPB	LITERALLY	20	
4	0004H	1	KPERBLOCK.	BYTE PUBLIC	21	
2			LF	LITERALLY		
2			LIT.	LITERALLY	2	3 10
7	0000H		MON4	PROCEDURE POINTER EXTERNAL(0) STACK=0000H		20
3			OFFSETW.	LITERALLY		
15	0004H	1	PARAM.	BYTE PARAMETER AUTOMATIC	16	17
11	0004H	1	PARAM.	BYTE PARAMETER AUTOMATIC	12	13
2			PROC	LITERALLY		
2			SECTORLEN.	LITERALLY		
			SHL.	BUILTIN	17	
			SHR.	BUILTIN	21	
3			SPTW	LITERALLY		
2			TAB.	LITERALLY		
2			TRUE	LITERALLY		

MODULE INFORMATION:

CODE AREA SIZE = 0060H 96D
 CONSTANT AREA SIZE = 0000H 0D
 VARIABLE AREA SIZE = 0005H 5D
 MAXIMUM STACK SIZE = 0008H 8D
 78 LINES READ
 0 PROGRAM ERROR(S)

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

END OF PL/M-86 COMPILATION

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

ISIS-II MCS-86 MACRO ASSEMBLER V2.1 ASSEMBLY OF MODULE SCD
 OBJECT MODULE PLACED IN :F0:SCD.OBJ
 ASSEMBLER INVOKED BY: :F0: SCD.A86

LOC	OBJ	LINE	SOURCE
		1	;
		2	;
		3	HP/M-86 2.0 with BDOS version 3.0
		4	Interface for PLN-86 with separate code and data
		5	Code org'd at 0
		6	October 5, 1981
		7	
		8	dgroup group dats,stack
		9	cgroup group code
		10	
		11	assume cs:cgroup, ds:dgroup, ss:dgroup
		12	
----		13	stack segment word stack 'STACK'
0000		14	stack_base label byte
----		15	stack ends
		16	
----		17	dats segment para public 'DATA' ;CP/M page 0 - LOC86'd at 0H
		18	
0004		19	org 4
0004 ??		20	bdisk db ?
0006		21	org 6
0006 ????		22	maxb dw ?
0050		23	org 50h
0050 ??		24	cmrv db ?
0051 ????		25	pass0 dw ?
0053 ??		26	len0 db ?
0054 ????		27	pass1 dw ?
0056 ??		28	len1 db ?
005C		29	org 5ch
005C (16		30	fcb db 16 dup (?)
??			
)			
006C (16		31	fcbl6 db 16 dup (?)
??			
)			
007C ??		32	cr db ?
007D ????		33	rr dw ?
007F ??		34	ro db ?
0080 (128		35	buff db 128 dup (?)
??			
)			
0080		36	tbuff equ buff
0080		37	buffa equ buff
005C		38	fcba equ fcb
		39	public bdisk,maxb,cmrv,pass0,len0
		40	public pass1,len1,fcb,fcbl6,cr,rr
		41	public ro,buff,tbuff,buffa,fcba
		42	
----		43	dats ends
		44	

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

LOC	OBJ	LINE	SOURCE
----		45	
		46	code segment public 'CODE'
		47	public xdos,mon1,mon2,mon3,mon4
		48	extrn plmstart:near
		49	
0000		50	org 0h ; for separate code and data
0000	EB4290	51	jmp pastserial
0003	434F5059524947	52	db 'COPYRIGHT (C) 1981, DIGITAL RESEARCH '
	43542028432720		
	313938312C2044		
	49474954414C20		
	52455345415243		
	4320		
0028	363534333231	53	db '654321'
002E	204D502F4D2D38	54	db 'MP/M-86 2.0, 10/5/81 '
	3620322E302C20		
	31302F352F3831		
	20		
0044		55	pastserial:
0014	9C	56	pushf
0045	5B	57	pop ax
0046	FA	58	cli
0047	8CD9	59	mov cx,ds
0049	3ED1	60	mov ss,cx
004B	8D260000	61	lea sp,stack_base
004F	50	62	push ax
0050	9D	63	popf
0051	E90000	64	jmp plmstart
		65	
0054		66	xdos proc
0054	55	67	push bp
0055	3BEC	68	mov bp,sp
0057	8B5604	69	mov dx,[bp+4]
005A	3B4E06	70	mov cx,[bp+6]
005D	CDE0	71	int 224
005F	5D	72	pop bp
0060	C20400	73	ret 4
		74	xdos endp
		75	
0054		76	mon1 equ xdos ; no returned value
0054		77	mon2 equ xdos ; returns byte in AL
0054		78	mon3 equ xdos ; returns address or word BX
0054		79	mon4 equ xdos ; returns pointer in BX and ES
----		80	code ends
		81	end

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

ASSEMBLY COMPLETE, NO ERRORS FOUND

ISIS-II MCS-86 MACRO ASSEMBLER V2.1 ASSEMBLY OF MODULE STAT86EX
 OBJECT MODULE PLACED IN :F0:ST86EX.OBJ
 ASSEMBLER INVOKED BY: :F0: ST86EX.A86 XREF DEBUG

LOC	OBJ	LINE	SOURCE
		1	\$title ('STAT.CMD - Compute Allocated blocks - External')
		2	name stat86ex
		3	;NOV 16 1981 - Doug Huskey
		4	
		5	cgroup group code
----		6	code segment public 'CODE'
		7	;
		8	assume cs:cgroup
		9	
0000		10	reset equ 0
001B		11	getalv equ 27
001F		12	getdph equ 31
		13	
		14	
		15	
		16	*****
		17	; A routine to access the CP/M-80 allocation vector
		18	; and compute the number of allocated blocks.
		19	; Upon entry the SP -> DSM (from the DPB).
		20	; The number of allocated blocks is returned in AX.
		21	*****
		22	
		23	public getall
0000		24	getall proc
		25	;
0000 55		26	push bp ;save stack offset for PLM
0001 B118		27	mov cl,getalv ;get os offset of allocation vector
0003 CDE0		28	int 224
0005 3BEC		29	mov bp,sp
0007 8B5604		30	mov dx,[bp+4]
000A 3BF3		31	mov si,bx
000C 1E		32	push ds ;save for exit
000D 06		33	push es
000E 1F		34	pop ds ;es = base of os (to get to alloc vector)
000F F5C207		35	test dl,7 ;check boundary
0012 7403		36	jz nbyte0
0014 33C203		37	add dx,8 ;need 8 more bits
0017		38	nbyte0:
0017 B103		39	mov cl,3 ;divide by 8 for number of bytes
0019 D3EA		40	shr dx,cl
001B 3BCA		41	mov cx,dx ;cx will contain number of bytes to examine
001D BA0000		42	mov dx,0 ;dl used to count allocated blocks
0020		43	nbyte:
0020 AC		44	lodsb
0021 3BD7		45	mov bx,cx ;save the byte count
0023 B90800		46	mov cx,8 ;examine 8 bits
0026		47	nbit:
0026 D0E8		48	shr al,1
0028 7301		49	jnb nbit1
002A 42		50	inc dx ;increment count of used blocks

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH

P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

LOC	OBJ	LINE	SOURCE
0023		51	nbitl:
002B E2F9		52	loop nbit
002D 3BCB		53	mov cx,bx ;restore cx
002F E2EF		54	loop nbyte
0031 1F		55	pop ds ;restore data segment before returning
0032 5D		56	pop bp ;restore stack offset
0033 8BC2		57	mov ax,dx ;ax = number of allocated blocks
0035 C20200		58	ret 2
		59	getall endp
		60	;
---		61	code ends
		62	end

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

XREF SYMBOL TABLE LISTING

NAME	TYPE	VALUE	ATTRIBUTES, XREFS
??SEG .	SEGMENT		SIZE=0000H PARA PUBLIC
CGROUP .	GROUP		CODE 5# 8
CODE .	SEGMENT		SIZE=0037H PARA PUBLIC 'CODE' 5# 6 61
GETALL .	L NEAR	0000H	CODE PUBLIC 23 24# 59
GETALV .	NUMBER	0018H	11# 27
GETDPB .	NUMBER	001FH	12#
NBIT .	L NEAR	0026H	CODE 47# 52
NBIT1 .	L NEAR	002BH	CODE 49 51#
NBYTE .	L NEAR	0020H	CODE 43# 54
NBYTE0 .	L NEAR	0017H	CODE 36 38#
RESET .	NUMBER	0000H	10#

ASSEMBLY COMPLETE, NO ERRORS FOUND

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____